2016

Application of active device authentication mechanisms in the human-machine interface of SCADA networks

Matthew Jeffrey Schlue
Iowa State University

Application of active device authentication mechanisms in the human-machine interface of SCADA networks

by

Matthew Jeffrey Schlue

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Co-majors: Computer Engineering and Information Assurance

Program of Study Committee:

Doug W. Jacobson, Major Professor

Manimaran Govindarasu

Mani Mina

Iowa State University

Ames, Iowa

2016

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

Getting to this point in my life would never have been possible on my own. I've read similar sentiments in acknowledgement sections before, and while agreeing with the idea, never truly understood the feeling until reaching this point in my life. I would have never have accomplished half of the personal, professional, and academic goals I have set out to achieve without the help of my close friends, family, professors, and mentors. This paper is dedicated to all those who pushed me to finish writing my thesis.

My parents have always believed in and encouraged me to pursue my dreams. Growing up, they worked tirelessly to provide me with every opportunity to learn, experience, and develop into the educated and well-rounded individual I am today. I'm eternally grateful for the role my sister has played in my life. Being an older sibling, she is always experiencing life one step ahead of me and takes the time to teach me the tips and tricks to succeed, and when I do stumble has always been there to lend guidance.

My professors, I thank them for seeing the potential in me and taking the time to mentor and teach me throughout undergraduate and graduate school.

My mentors, Matthew and Ben. They guided me through life as a graduate student, and even now are helping review my thesis as I write this.

And lastly my friend Laura, who always encourages me in whatever endeavor I'm working to accomplish, and inspires me to constantly strive to become the best version of myself.

ABSTRACT

Supervisory Control and Data Acquisition (SCADA) systems are a type of Industrial Control System (ICS) that both monitor and control the critical infrastructure that delivers manufactured goods, water, and energy. These systems are responsible for supervising everything from natural gas valves to electric substations. For the past half century, SCADA and ICS networks have been proprietary, closed systems, entirely contained within a private network. Their security was derived from air gap networking, physically isolating these systems from the Internet. However, system operators are increasingly opting to connect their control systems to Internet or corporate intranet networks in order to substantially reduce operating costs and improve reporting capabilities. This architecture change has given rise to a new and poorly understood class of risk.

In this work, we examine how a security concept known as Active Device Authentication can be applied to the SCADA system threat model. As our contribution, we develop a software tool known as *Gatekeeper* that wraps Active Device Authentication capabilities around existing, weaker authentication mechanisms present in off-the-shelf HMI software written in Java. This work aims to provide the reader with a stronger understanding of the concept of Active Device Authentication, and how it can be deployed into legacy, proprietary, or mission-critical environments to enable additional security controls without risk of impacting the underlying systems' reliability.

CHAPTER 1. BACKGROUND

1.1 SCADA Control System Security

Supervisory Control and Data Acquisition (SCADA) systems are defined as systems used to control dispersed assets using a centralized system to collect data and control corresponding components, Stouffer et al. (2011). These systems are rapidly being integrated into the public Internet. On December 7, 2007 the president signed the Energy Independence and Security Act into law, Independence (2007). Under this new act, the Department of Energy, along with several other organizations and agencies, were given the monumental task of modernizing the US power grid. This has lead to creation of "third generation SCADA", or Internet connected critical infrastructure systems, Cai et al. (2008). As these systems become increasingly interconnected with the rest of the world, their attack surface and the corresponding security implications drastically change.

Over the past decade, the amount of computing resources, number of attack tools, and network speeds are all increasing, while the IPv4 address space is not. In September of 2013, HD Moore, the creator of the Metasploit security toolset, stated that "The IPv4 internet is more of a kiddie pool than this nebulous unknowable thing anymore" Moore (2013). Moore correctly observes that the modern attacker has the ability to enumerate specific vulnerabilities and services across the entire IPv4 space in an alarmingly short timespan. It is no longer a question of whether misconfigured and vulnerable pieces of a critical infrastructure will be discovered, but when they will be found (and by whom). For example, the Zmap project from the University of Michigan is capable of scanning the entirety of the IPv4 address space in under an hour from a single machine with a gigabit internet connection, Durumeric et al. (2013). Obscurity can no longer be an assumed defense mechanism of Internet connected

infrastructure. These network-facing systems are likely to be probed on a regular basis, even daily, Cai et al. (2008). SCADA systems are, for the moment, surprisingly under-equipped to handle this new threat, which could, and likely will, have costly real-world ramifications.

In this work, we will cover the current security posture and implications of third generation ICS and SCADA systems, which are characterized by having Intranet connectivity. We will discuss common configurations found in SCADA and industrial control systems, security issues of these configurations, including current protection and mitigation strategies recommended by ICS-CERT and other authorities. This paper will attempt to show that these recommended security controls, while valid, can still be circumvented.

Next we propose a novel concept that will further secure a critical element of industrial control systems, the human-machine interface. Our research will demonstrate protecting the HMI component by actively collecting data about end user's systems accessing the SCADA network. Since legacy, proprietary, HMI systems were not designed with security in mind, authentication logic must be added as a layer on top of the original software system. By injecting logic to actively fingerprint the client system we can gather data that has been previously unattainable in traditional intrusion and anomaly detection systems. The enriched information obtained from a connecting client's system is used to more accurately determine the validity and authorization of the connecting device. We will refer to this technique as *active device authentication*.

As a part of our evaluation, we show that even when authentication credentials for an operator are compromised, a system is still able to accurately protect the HMI system from unauthorized access. This is achieved by matching a whitelist of characteristics about a client's system, such as the MAC address, screen resolution, or operating system. This information, now provided through the application layer, was previously unavailable to use for authentication. An *active device authentication* mechanism, when combined with industry standard security practices, will provide a greatly increased defensive posture without hindering system performance or accessibility. This work also demonstrates that current security controls and procedures alone are not enough to protect against a persistent and determined hacker, or nation-state attack.

While the solution outlined in this paper is inimitable; the problem explored in it is hardly new. At its core, there is a dispute of usability, performance, and uptime versus the security and integrity of a system. In the SCADA world, for example, it is not uncommon for a firewall to fail open when receiving more packets than it can process, to prevent time sensitive and mission critical data from being dropped. As more layers and complexity are added to this system or network, the possibility arises of it becoming less user-friendly, and potentially less functional. Many control system networks rely on real-time data transmission and computing, which cannot withstand such outages caused by stringent security measures. In fact, a brief glance into the priorities of a SCADA or HMI network operator will reveal their values to typically be the exact opposite of the traditional Confidentiality, Integrity, and Availability model of security, Wang and Lu (2013). The challenge then lies within security professionals to provide thorough, layered, and robust security for some of the most critical information systems in the world without impeding performance or availability.

1.2 Current Landscape of SCADA Networks

The topological layout of control system networks varies widely, however there are a few common consistencies in the design. NIST and ICS-CERT have published several detailed ICS network designs. For the context of this work, these theoretical network implementations will be used as a basis to demonstrate the capabilities of an active defense approach to authentication and authorization. First we will describe several of these common network architectures, followed by the importance and challenges of protecting them. The goal of this section is to show that while there are currently many protection mechanisms on a power system network, in many cases it is still trivial for an attacker to reach an HMI console. Finally, we will discuss the importance of placing an additional protection mechanism, proposed later in this work, closer to the information it is protecting. This layered protection will greatly add to the overall effectiveness of a comprehensive security strategy.

The network layout in Figure 1.1 includes several firewalls between the sensitive internal SCADA network and the public Internet. At minimum one firewall system will be incorporated into most designs.

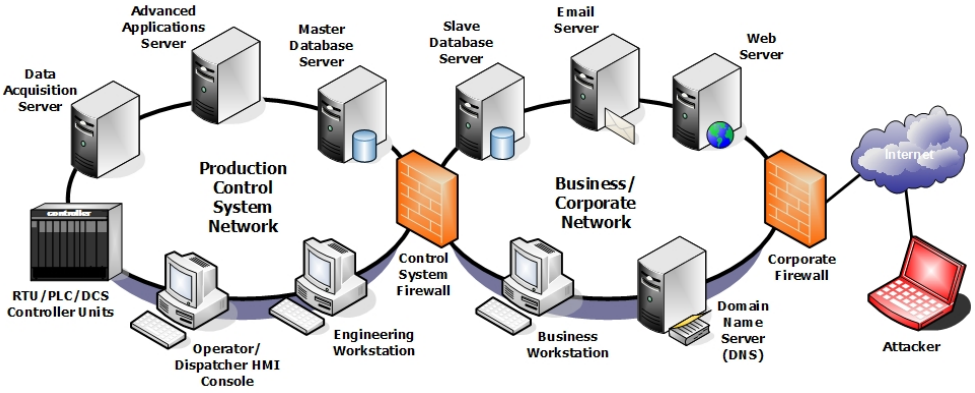

Figure 1.1 Two Firewall SCADA Network Architecture, US ICS-CERT

The corporate network commonly includes several common systems such as email, web services, and employee workstations that are on many typical office intranet networks. It is important to note that the information systems in the business network will conform to separate uptime, maintenance, and security standards from the internal control system network. At first glance, this network may seem fairly secure. With two firewalls, network administrators are able to create multiple sets of access control lists (ACL) to block malicious traffic and easily segment access to sections of the network. Without direct access, this layout creates a considerable challenge for an attacker looking to access the control system network. However, it is not immune to penetration.

An attacker will begin intrusion by targeting services advertised publicly through the corporate firewall. Next, the attacker will attempt to compromise a machine on the corporate network, whether by phishing or some other mechanism. Once this task has been completed, the attacker will begin the scanning and enumeration process in order to understand the internal network. An adversary will be able to pivot to weaker systems once inside the network, allowing greater access and the ability to find and compromise a machine that has access to systems and services on the secondary control system network. For example, an attacker can compromise an insecure application on the corporate web server; from that entry point, the attacker may scan for an outdated operating system to exploit. After finding and compromising one or more systems, the attacker can pivot into a system with access to the production control system LAN.

While this scenario appears difficult to exploit, it is in fact much simpler than one might expect. The corporate systems that connect to the Internet provide significant attack surface. Web servers, due to the inherent nature of being public internet facing, are one of the easier systems on a network to exploit. Custom web applications will likely provide plenty of opportunities to gain access through one of several common and easily detected programming errors. Employees using the corporate network also pose one of the greatest security risks. A joint study between Google and UC San Diego showed the astonishing effectiveness of targeted phishing attacks. Multiple assessments indicated an average 13.7% success rate against selected victims, with some assessments returning as high as 45%. What is even more astounding is the rate at which attackers utilize access obtained through phishing. As part of background research conducted during the same study, it was found that 50% of the credentials obtained were utilized by attackers within 7 hours, with 20% being consumed within 30 minutes Bursztein et al. (2014). A similar study with USB flash drives was conducted with an estimated attack success rate of 45%-98% with the first drive being activated in less than 6 minutes, Tischer et al. (2016). In fact, Stuxnet, one of the most widely cited cyber-physical attacks leveraged traditional USB based malware self-replication logic to jump between air gapped networks, Langner (2013). When the preceding facts are considered, it becomes increasingly hard to argue that a simple and common network security setup, such as the one depicted in Figure 1.1, can be sufficient for use in SCADA systems.

In the second example of an ICS network layout, the topology is very similar to Figure 1.1 with one very significant difference. The internal production control system network is accessible through the firewall using a virtual private network connection or VPN. Similar to the previous example, an attacker can take several approaches to compromising the necessary credentials to login to the VPN application. For the scope of this paper we will make the postulation that this breach in security is possible.

With complete access to the internal ICS network through VPN, the attacker's system may as well be physically plugged into the same network switch as other equipment. Many of the attacks, network scans, and exploits against the Human Machine Interface that would be stopped by the ICS firewall in Figure 1.1 are now possible in Figure 1.2. One of the simplest

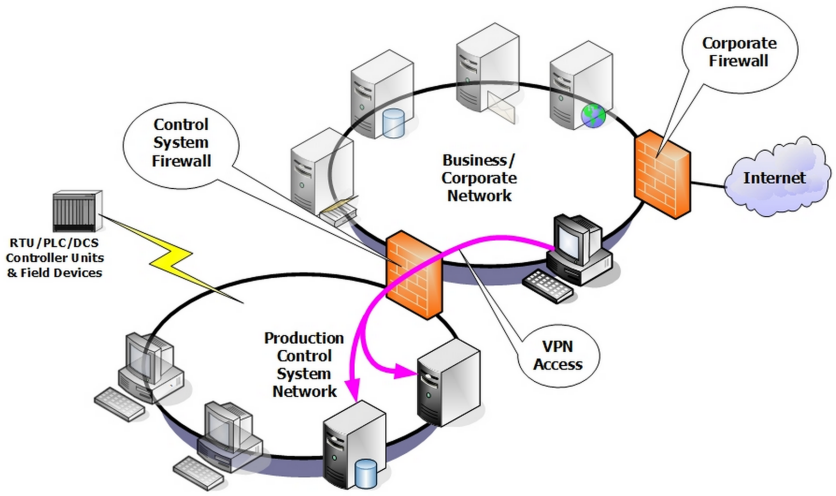

Figure 1.2 Firewall System with VPN Access, US ICS-CERT

attacks to accomplish is ARP cache poisoning. With this attack, devices on the LAN are tricked into thinking an attackers laptop is the correct destination for network packets. This can be utilized to intercept and read all the packets destined for the Human Machine Interface. This attack has the capability to block important or time sensitive communications and capture credentials being transmitted through the network, Wang and Lu (2013). At this stage the malicious actor is able to freely access the HMI console and manipulate data on the ICS network without any restrictions, and with the ability to utilize the VPN have guaranteed access around any perimeter security devices.

In Figure 1.3, the ICS network topology allows vendors to remotely assist control center engineers and IT personnel with upgrades. In this scenario remote substations are attaining access to the LAN with dial-up modems, and by VPN in more modern networks, ICS-CERT (2016). As witnessed in the attacks against the Target Corporation, their most recent major breach was caused by an HVAC vendor's remote access to their internal network, Yadron (2014). A vendor is not always bound with the same stringent policies and security controls as a critical utility. This makes it possible, and even easier for an attacker to target the vendor to steal login credentials to a remote SCADA network.

In each of these models is imperative to convey the significant level of risk that is inherent in maintaining a complex information systems network. A company requires a large amount of

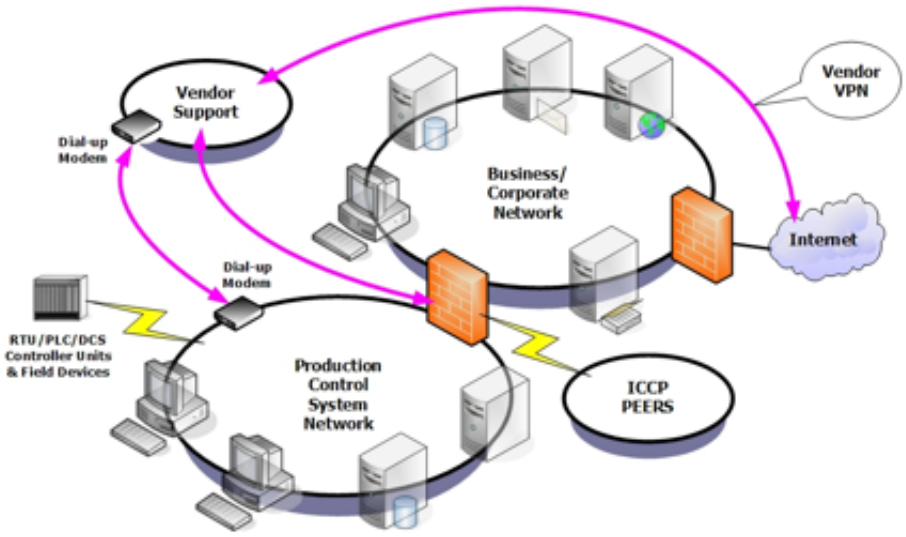

Figure 1.3 Network dialup access, and vendor support, US ICS-CERT

capital resources, engineering hours, and software tools to develop a secure network. After an immense amount of effort, an attacker only needs to find a single unpatched operating system, misconfigured firewall, or vulnerable web application to gain complete access to an ICS network.

> The interaction between defenders and attackers in cyberspace is highly asymmetrical. A network administrator must be constantly on guard, and make a large investment in software and appliances to secure the network from attack. Information systems are highly complex, and understanding of their internals and configuration is often highly compartmented. In many large organizations, a Chief Information Security Officer (CISO) must delegate security monitoring of various systems to subordinates who are experts in those subject areas. In larger networks, it might not be possible for a single network administrator, or even a small team, to keep up with the constant stream of possible vulnerabilities to defend against. Blakely (2012)

Add to this situation the stated primary objective in SCADA of availability over all else, and it quickly becomes an uphill battle.

1.3 Change Management and Patching in SCADA Systems

SCADA networks have very different maintenance and upgrade policies than a standard corporate network. Change management is a very difficult undertaking with these mission-critical, high uptime systems. Architectures between competing ICS technologies are often incompatible, and not interoperable. This creates a situation where patches for software bugs, security flaws, or even new features rely heavily on the vendor's discretion and capabilities to fix. For these reasons many SCADA and ICS systems are purchased with a vendor agreement for continued support. However, if a vendor is unable to provide a security patch for a particular SCADA system in a timely manner it will likely have to remain online and vulnerable for a prolonged period of time.

Applying upgrades to equipment on the control system network while minimizing outages can be a complex process that often involves IT, security personnel, SCADA engineers, and the vendor. SCADA equipment is both expensive and designed for a very specific purpose, therefore the life cycle of these systems is intended to be much longer than an average IT system. Tables 1.3 and 1.3 show the typical lifecycle being designed to last between 15 to 20 years. As a result of this paradigm the operating system powering the SCADA software will be utilized well past the time in which it's vendor is providing security patches. It is not uncommon for example to still be using Windows XP despite Microsoft no longer officially offering support, Stouffer et al. (2011).

According to the NIST report, a survey of a critical infrastructure network at a major power provider revealed that their SCADA and ICS networks were connected, by some means, into their corresponding corporate network. It can be generalized that a corporate network is likely not up to security standards required to operate safety-critical systems. The requirements and goals of an IT network are drastically different than a SCADA network. It was also noted by NIST that those in upper management were not aware of the interconnectivity of business and control system networks, Stouffer et al. (2011). It is inherently difficult to secure a network, when those making decisions on how this network should be designed and protected are unaware of the architecture and the related security implication of their interconnected systems.

The implications of SCADA technology described above create a myriad of security concerns. Network access to outdated systems, such as Windows XP, which no longer have vendor support, generates a grave threat, as any newly discovered exploits will never be patched by the official vendor. Any software bugs in the HMI application are completely dependent on the ICS vendor providing timely updates.

Table 1.1 Summary of IT and SCADA System Differences, NIST Guide

Category	**IT Systems**	**Industrial Control Systems**
Performance	• Non-real-time • Response must be consistent • High throughput is demanded • High delay and jitter may be acceptable	• Real-time • Response is time-critical • Modest throughput is acceptable • High delay and/or jitter is not acceptable
Availability	• Responses such as rebooting are acceptable • Availability deficiencies can often be tolerated, depending on the system's operational requirements	• Responses such as rebooting may not be acceptable because of process availability requirements • Availability requirements may necessitate redundant systems • Outages must be planned and scheduled days/weeks in advance • High availability requires exhaustive pre-deployment testing • Human safety is paramount, followed by protection of the process
Risk Management	• Data confidentiality and integrity is paramount • Fault tolerance is less important momentary downtime is not a major risk • Major risk impact is delay of business operations	• Fault tolerance is essential, even momentary downtime may not be acceptable • Major risk impacts are regulatory non-compliance, environmental impacts, loss of life, equipment, or production
System Operation	• Systems are designed for use with typical operating systems • Upgrades are straightforward with the availability of automated deployment tools	• Differing and possibly proprietary operating systems, often without security capabilities built in • Software changes must be carefully made, usually by software vendors, because of the specialized control algorithms and perhaps modified hardware and software involved

Table 1.1 (Continued)

Category	**IT Systems**	**Industrial Control Systems**
Change Management	• Software changes are applied in a timely fashion in the presence of good security policy and procedures. The procedures are often automated.	• Software changes must be thoroughly tested and deployed incrementally throughout a system to ensure that the integrity of the control system is maintained. ICS outages often must be planned and scheduled days/weeks in advance. ICS may use OSs that are no longer supported
Component Lifetime	• Lifetime on the order of 3-5 years	• Lifetime on the order of 15-20 years
Managed Support	• Managed Support	• Service support is usually via a single vendor

CHAPTER 2. RELATED WORK

2.1 Human Machine Interface Risk Assessment

Similarly with network designs, HMI software also varies widely by vendor. HMI software has gone through significant changes in recent years to be more inline with the objectives of third generation ICS networks. Early HMI implementations often had a specific computer system with proprietary software installed, which was directly connected to the ICS network. During this time, being inside an air-gap network design inherently protected the critical infrastructure systems and HMI console, Urias et al. (2012). Many SCADA protocols and ICS software suits at this time were designed without implementing basics security measures. For example, the original design of DNP3 does not include any authentication to send or receive packets as part of the protocol. This allows an attacker to easily craft a custom DNP packet or replay packets captured across a LAN. After gaining initial access to an internal network, an attackers work to compromise functionality in a SCADA environment becomes trivial. However, at that time security was not, and probably did not need to be the primary objective to keep systems online and safe in the air-gap topology. While the impact of a breach in this isolated scenario is still very high, the risk would remain low, as it requires physical access to the system or network. In fact before 2001, half of the incidents reported to affect operation in SCADA systems were attributed to human error from workers on site, Urias et al. (2012).

Over the past decade, increased broadband penetration, the industry trend of networking remote systems, and demand by both engineers and business units for remote access to information within ICS systems has prompted a change in connectivity options, Stouffer et al. (2011). With systems being rapidly connected to the Internet an entire new realm of security challenges and breaches were introduced.

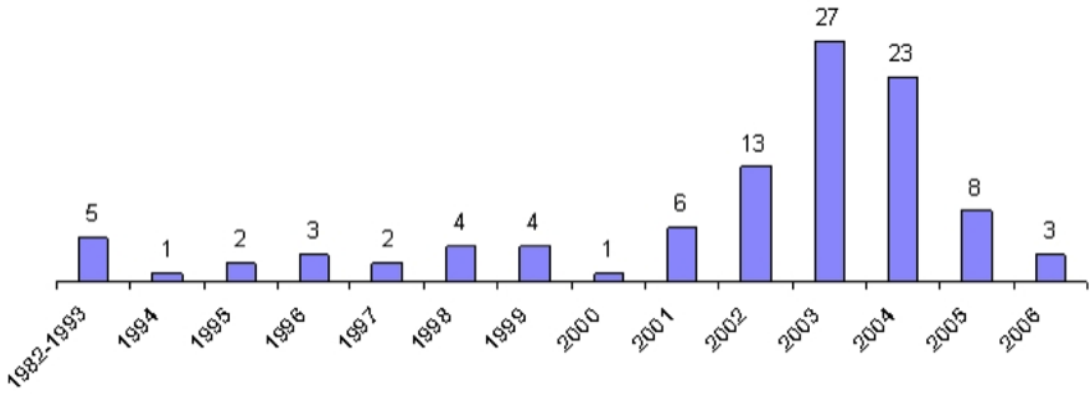

Figure 2.1 Industrial Security Incidents by Year, NIST

The Human Machine Interaction software can be a dedicated machine in a control center with firewall rules to allow remote access from business or VPN connections. The HMI console also has the possibility to be a software package installed on a company laptop, which can access the information anywhere inside the corporate, control system, or VPN network. Most notably, and the focus of this research, modern HMI software is able to be setup to allow access from a web browser by any information system connected to the Internet, Stouffer et al. (2011). This final option provides the most flexibility and accessibility to engineers interacting with internal systems, but with a huge security tradeoff. The risk and impact from vulnerabilities in web based HMI software is drastically amplified by the increased attack surface of internet-wide connectivity.

2.2 Intrusion Detection Systems in SCADA Networks

Traditional IT approaches can be implemented into a SCADA network in an attempt to improve security and prevent unauthorized access. This section will explore how intrusion detection systems (IDS) operate on a SCADA network. It will cover what type of data can be gathered at the network layer by an IDS, and how this data can be used to enhance security around and HMI application. Finally this section covers what gaps still exist when protecting a web application when using an IDS system.

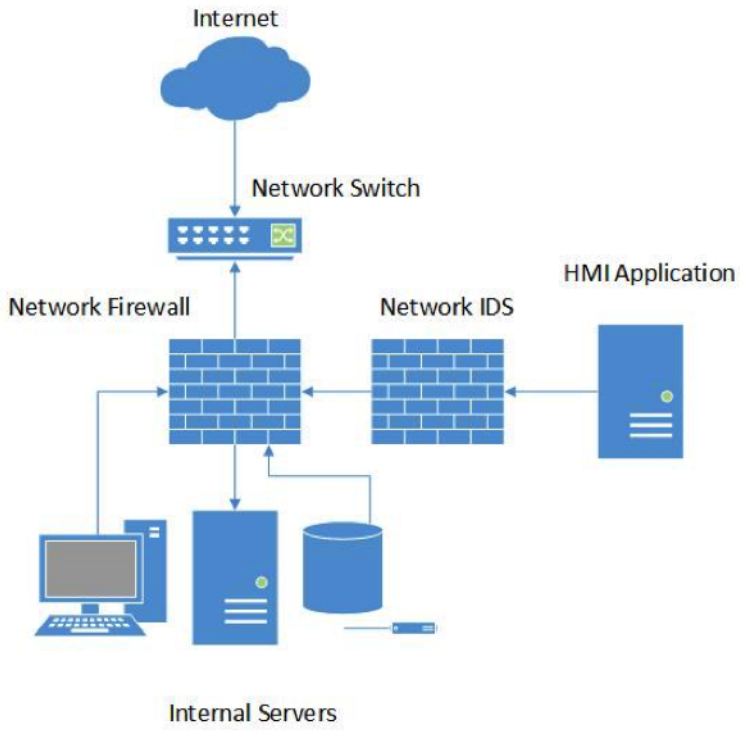

Figure 2.2 Typical Network Topology for IDS Implementation

To utilize an IDS to its full potential when protecting a specific asset, such as an HMI application, it should be topologically placed close to that asset on the SCADA network. This allows the system to be tuned to specifically watch and filter traffic to specifically protect an application. The more resources contained behind the IDS, the higher probability of being affected by a false negative.

Figure 2.2 shows a typical setup where an IDS system is used to protect a specific web application, in this instance an HMI web application. The firewall in this diagram is placed behind the main router for the network. It will protect systems based on an access control list or ACL. This list contains whitelisted IP addresses that are allowed to send traffic, and the ports through which they can route traffic. This provides a first step and basic security to the network, however its effectiveness is limited to information contained in the network layer of the OSI model. The IDS system in Figure 2.2 is placed directly in front of the HMI web application. Previously, we highlighted how attacker will often target a separate system in a network and use that access to pivot into attack an HMI application from an internal

network address. This IDS placement prevents that by further isolating the web server from both external attacks and intrusion attempts from internal systems on the network, Rehman (2003).

The intrusion detection system on the network can be tuned to detect signatures of malicious traffic such as port scanning, a sudden increase in network packet rates that may result in denial of service, or even as simple as sending alerts based on the network protocol used, Rehman (2003). In an IDS these events are categorized using rule sets. Rules are a list of signatures that match known traffic or attack patterns. Once a signature match occurs the IDS will perform one or more of the following actions: blocking traffic from the source IP, alerting a system administrator, or logging the event, SANS (2008).

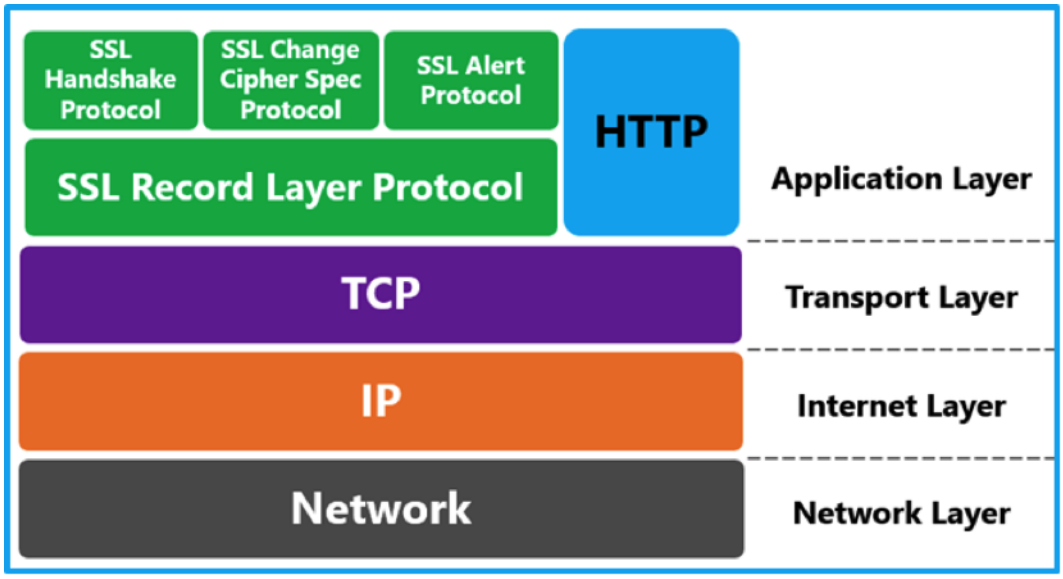

Figure 2.3 Protocol Stack for SSL Traffic, MSDN (2016)

While these security methods by an IDS can aid in the detection of attack traffic on a network they have several significant drawbacks when protecting a web based HMI application. As best practice a web application should be protected with an SSL certificate. This is especially important when the web server transmits important data in a SCADA network. Having the HMI applications traffic contained inside an encrypted SSL connection will prevent eavesdropping or man in the middle (MiTM) attacks. That encrypted channel will also prevent the IDS from reading any data inside the application layer. This setup is visualized

in Figure refssl-protocol-stack, as we can see the IDS will only be able to read the routing information in the first three networking layers. If the a legitimate user's credentials are stolen through one of the attack scenarios outlined in Chapter 1, and the malicious actor is able to spoof or connect from an acceptable IP address there is very little can IDS can do to prevent the attack, SANS (2008). In the next chapter we will discuss inserting additional data into the encrypted application layer, and describe how this data can be utilized to prevent an attack that would be permitted by a standard IDS.

According to a report by NIST, a survey of a critical infrastructure network at a major power provider revealed that their SCADA and ICS networks were connected, by some means, into their corresponding corporate network. It can be generalized that a corporate network is likely not up to security standards required to operate safety-critical systems. The requirements and goals of an IT network are drastically different than a SCADA network. It was also noted by NIST that those in upper management were not aware of the interconnectivity of business and control system networks Stouffer et al. (2011). It is difficult to secure a network, when those making the decisions on how this network should be designed and protected are unaware of the architecture and the related security implication of their interconnected systems. Given its level of access and ability to remotely interact with critical infrastructure systems, ICS-CERT recognizes the HMI console as being one of the most valuable targets for an attacker to obtain access ICS-CERT (2016). Defending an HMI system must also be given the same level of diligence.

CHAPTER 3. ACTIVE DEVICE AUTHENTICATION

As discussed previously, an Internet facing HMI console to access a utility's critical internal systems presents a huge security challenge, especially when the software is created and maintained by a third party. This often leads IT administrators with little or no ability to provide additional security measures to the HMI application. Existing solutions are applied at the IP network layer or above by attempting to put a firewall network boundary, or IDS in front of the application. In the latter half of Chapter 1 we discussed how attackers are still able to remotely gain access to applications, creating a need for further security beyond what is currently available.

This chapter proposes a solution for access control and device authentication to a legacy HMI console providing inadequate security or completely lacking security measures. The proposed solution shows how to prevent unauthorized access from an attacker, even when a legitimate username and password have been compromised through the use of multiple authentication challenges.

3.1 A Solution for Legacy Device Authentication

Our proposed device authentication technique involves injecting code into an HMI application to add client fingerprinting and client-server authentication logic before allowing a client to access sensitive data on the SCADA network. There are two components that comprise the additional security measures, injected client-side code that is run to gather profile data about a user's system, and the server-side logic, which we call *Gatekeeper*, that checks the profile data against a whitelist of known client profiles and reconfigures a firewall to allow the client access to the SCADA network. Optionally, *Gatekeeper's* whitelisting strategy could be replaced with

an adaptive machine learning approach to incorporate networks that are highly dynamic and prone to change.

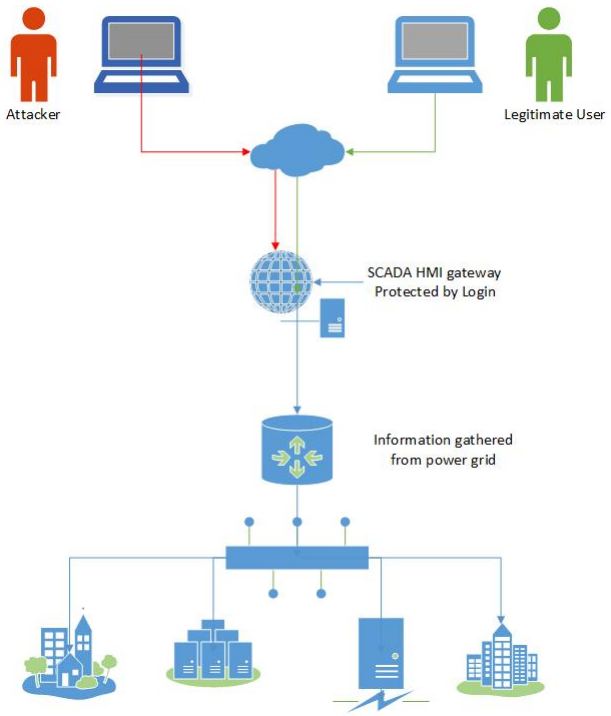

Figure 3.1 Unprotected HMI Application

In Figure 3.1 and Figure 3.2 both the malicious actor and the legitimate user are able to directly access the HMI application on an Internet facing network. Figure 3.2 demonstrates the modified HMI application interacting with *Gatekeeper*. Note that the modified client is only facilitating data collection for *Gatekeeper*. *Gatekeeper* maintains authority to fulfill or reject client side requests with the SCADA network. If a malicious actor attempted to remove or disable the HMI modifications *Gatekeeper* would have no reason to allow the client access through the firewall. It is possible that an attacker would try to fake the client profile information collected by the modified HMI application, but by increasing the number of profile data points it becomes increasingly harder for an attacker to provide valid profiles.

The injected software security layer examines distinct sets of the client system information for a connecting client that when combined together, creates a unique connection profile. This connection profile is then evaluated by Gatekeeper's whitelist and, if need be, various authen-

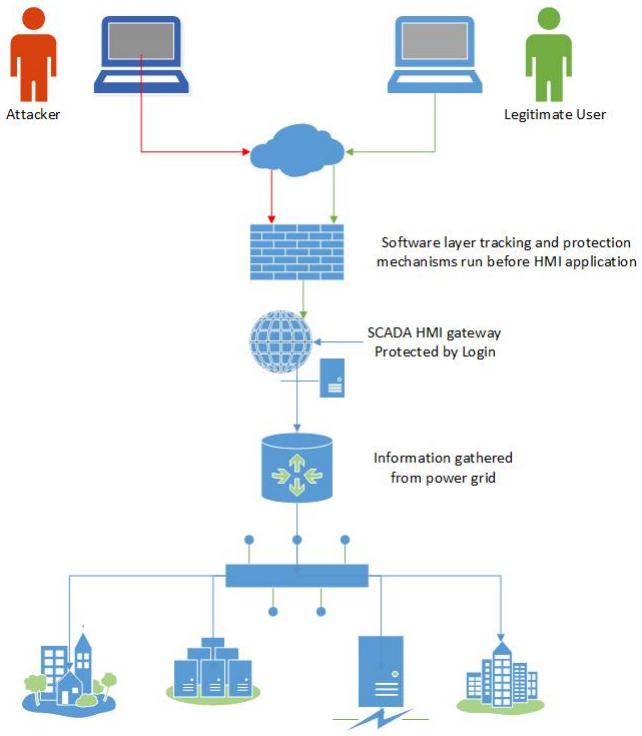

Figure 3.2 Protected HMI Application

tication challenges (password or various two factor based authentication methods) are issued to the modified HMI application to provide a means to grant access when new profiles are detected. In this work we experimented with the following actively collected profile data shown in Table 3.1.

Figure 3.3 shows the additional protection mechanisms added to the HMI application. The HMI application, while still served to the client over the network in the same manner, is now wrapped with logic to send client system profile information and relay user responses to *Gatekeeper's* authentication challenges. If a connected client fails multiple authentication challenges, *Gatekeeper* notifies the network administrator of the suspicious activity.

Table 3.1 Data Points Gathered from Client System

Metric	Information about Metric	Caveats about Metric
Username	Name used to login to the remote computer	None
Operating System	Version information about the client's system	None
Java Version	Used to run the provided JAR file	This can change with system updates
MAC Address	Unique hardware identifier of the NIC	None
Hostname	End user's system hostname	None
Screen Resolution	Screen size ex: 1440x900	Users may change their screen size
IP Address	The connecting clients IP address	IP addresses will change as users travel
Wireless MAC	Wireless hardware address	Not all devices have wireless

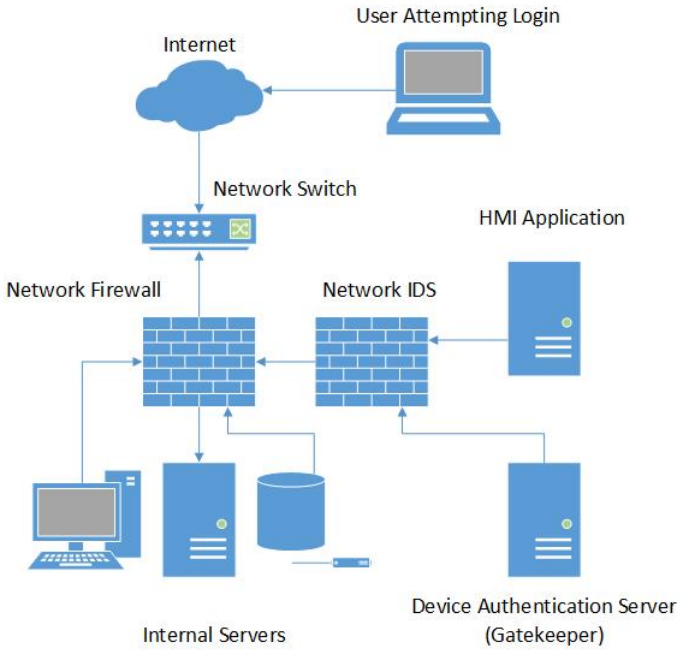

Figure 3.3 Device Authentication Server in SCADA Network

CHAPTER 4. EVALUATION

This chapter evaluates a proof-of-concept implementation of the *active device authentication* technique proposed in Chapter 3.

Several example HMI systems are freely available online for trial and testing, and can be used to demonstrate our proposed security measures. At the time of this writing SourceForge, one of the most popular open source code repositories, showed Java applets being one of the most utilized language platforms in writing a web application based HMI console. Cisco's SCADA Honeynet project also leverages Java applets, which partially confirms this trend, Pothamsetty (2016). While the solution in this paper can be deployed with for many other programming languages and frameworks, we will focus on evaluating the technique for HMI applications written in Java.

Java is a popular choice for Human Machine Interaction software as it is cross-platform, and can be used to provide a robust, configurable graphical user interface to display status information and allow operator interaction. Prior to 2015, when most major browsers collectively disabled execution of even signed Java applets due to security concerns, Java applets served over a network through a web page were a popular choice for connecting clients on the network. This method also accommodated the trend of SCADA system operators being able to perform tasks remotely. Access to the application is as simple as navigating to a URL in a browser that supported Java. Since applets had be deprecated by most browsers during our evaluation we ran the HMI Java applet as a Java application to proceed with the evaluation. Java applets can be trivially modified to run as normal Java applications, so it is likely that many legacy Java applets are still running today, despite legitimate security concerns.

4.1 Gatekeeper Implementation

It is common knowledge in the web application security community that client-side validation, such as JavaScript checks, are not enough to stop an attacker. Any application presented to a user has the potential to be modified or reverse engineered before use. Authentication mechanism's that rely on the integrity of client-side application checks can be easily invalidated. For evaluation in this work, a server side authentication application named *Gatekeeper* was designed (outlined in Chapter 3). Knowledge and device challenges presented to a user have to be verified by the *Gatekeeper* software before a system is allowed access to a data source for the HMI application.

Gatekeeper is a Python application written on top of the Flask framework. It provides API endpoints to receive system profile information from a client attempting to run an HMI application. The information received is checked against a whitelist of known profiles. If the profile is not found in the whitelist, the session is issued an authentication challenge before it is allowed to connect to the network. In the current implementation, authentication challenges consist of either a username and password pair or an SMS based two-factor authentication.

The complete source code to *Gatekeeper* is available at: https://github.com/mschlue/gatekeeper.

4.2 HMI Application Modifications

The Java application for a HMI system is comprised of a series of compiled class files and archived in a JAR file. A typical distribution method is to deploy the JAR file to a web server, to be downloaded and executed by the user.

Gatekeeper expects clients to send a specific list of system profile information. Since the legacy HMI application does not include logic to interact with *Gatekeeper* this logic must be injected into the compiled proprietary HMI application. In this evaluation, JReFrameworker https://github.com/benjholla/JReFrameworker, a bytecode manipulation framework was used to inject the system profiling and GUI logic for responding to *Gatekeeper* authentication challenges.

4.3 Active Device Authentication Proof-of-Concept

During our survey of open source SCADA applications we discovered a sample interface for a hypothetical SCADA application released by GenLogic available at: http://www.GenLogic.com/java2/scada_viewer.html. The application is typical of larger complex software projects in that it uses multiple libraries, static resource files, and contains complex user interactions.

To begin our evaluation we first compiled the application (to simulate a lack of source code, which is the case for most legacy proprietary HMI applications). We then injected the system profiling logic into the startup routine of the compiled application using JReFrameworker. JReFrameworker allowed us to quickly create a project containing the source code of the logic we wanted to inject into the compiled application. The system profiling logic injected into the GenLogic SCADA demonstration application in this evaluation collects the username, operating system, Java version, mac address, hostname, and IP address of the client machine. In addition to injecting system profiling logic we also injected user interface code to facilitate responding to *Gatekeeper* authentication challenges.

In Figure 4.1 we can observe that the unmodified Java application immediately starts and is available for interaction from a user. This is an example of an application that is likely to be running on an internal network. The application itself does not provide any protection mechanisms and relies on network and system administrators to create access policies and firewall rules for protection. As mentioned in Chapter 1 these rules can be defeated by an attacker, and we therefore need to inject application layer active device authentication.

When running the updated HMI application the collected information is sent in a JSON over an HTTP POST request to the *Gatekeeper* authentication server. An example JSON payload sent to the application includes: {*"mac": "00:50:56:8e:4c:67", "os": "Mac OS X", "java_version": "1.8.0_60-b27", "hostname": "LS2-00391", "username": "matthewschlue"*}. In the first transaction all of the supplied information from the modified HMI application is contained in the whitelist of the *Gatekeeper* authentication server. The outcome of this is a successful authentication attempt that grants the user access. Figure 4.2 shows the device authentication transaction from the point of view of the authentication server's logs.

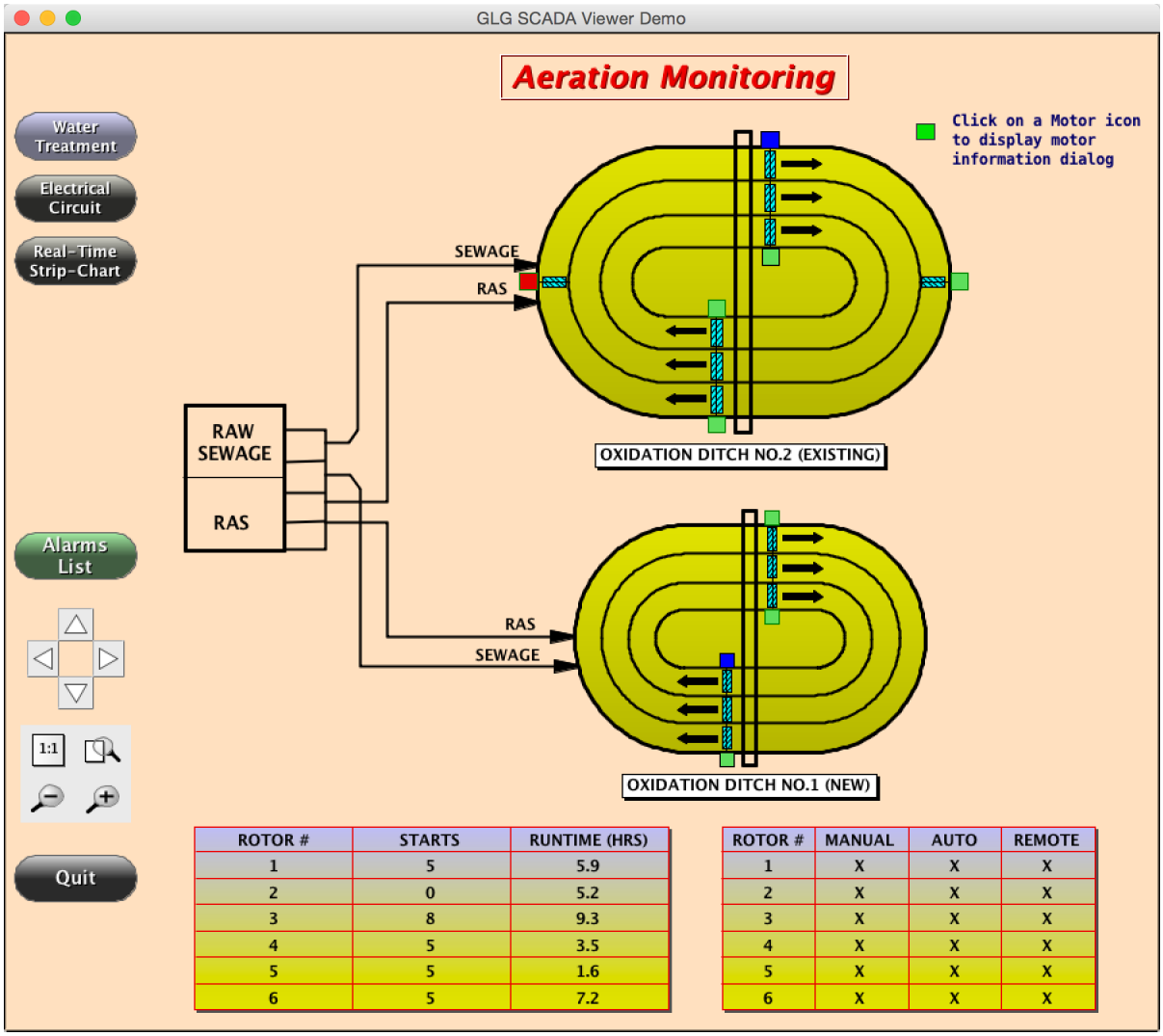

Figure 4.1 Unprotected HMI Application - GenLogic

In the second payload sent, a bad hostname was passed into the authentication server. Figure 4.3 shows the auth server individually checking each metric passed to the server. When the hostname check failed, the remaining checks immediately stopped and the connected user was prompted for a two-factor challenge. In this case the two-factor challenge was a unique one time link sent by SMS to the user's mobile phone shown in Figure 4.5. When the challenge received the correct response shown in Figure 4.4 the user was allowed to connect to the network.

The third scenario an attacker does not have the proper device profile and fails the hostname check against their machine. It is important to note that the values submitted to the authentication server in this scenario may be spoofed if the attacker had the skill to reverse engineer the applet before running it. Even with spoofed values the attacker still fails one of the system metrics and is prompted for a two-factor authentication code. Upon being unable to provide the second authentication factor, the attacker is given a 403 HTTP response, and the *Gatekeeper* server refuses to allow access to the offending system's IP address.

```
WARNING:root:starting web service
Checking client data payload: {u'username': u'matthewschlue', u'java_version': u'1.8.0_60-b27'
Client passed the check for username!
Client passed the check for java_version!
Client passed the check for mac!
Client passed the check for hostname!
Client passed the check for os!
IP 69.5.131.1 added to the authorized white list
69.5.131.1 - - [2016-04-12 01:24:00] "POST /auth HTTP/1.1" 200 149 0.004738
```

Figure 4.2 Client With Passing Host Metrics

```
WARNING:root:starting web service
Checking client data payload: {u'username': u'matthewschlue', u'java_version': u'1.8.0_60-b27',
Client passed the check for username!
Client passed the check for java_version!
Client passed the check for mac!
Client failed the check for hostname
Sending two factor challenge
generated two factor token 605968
Sent SMS message to client!
69.5.131.1 - - [2016-04-12 01:35:02] "POST /auth HTTP/1.1" 403 256 0.765681
```

Figure 4.3 Client Prompted For Two-Factor Authentication

From the point of view of the HMI user, there will not be any perceived differences when starting the application if their system profile is whitelisted by gatekeeper. If a host check is

```
Fetching tokens from Redis
Token validated, adding IP address 69.5.131.1 to whitelist
69.5.131.1 - - [2016-04-12 01:44:54] "GET /2fa?token=605968 HTTP/1.1" 200 150 0.002029
```

Figure 4.4 Successful Two-Factor Authentication

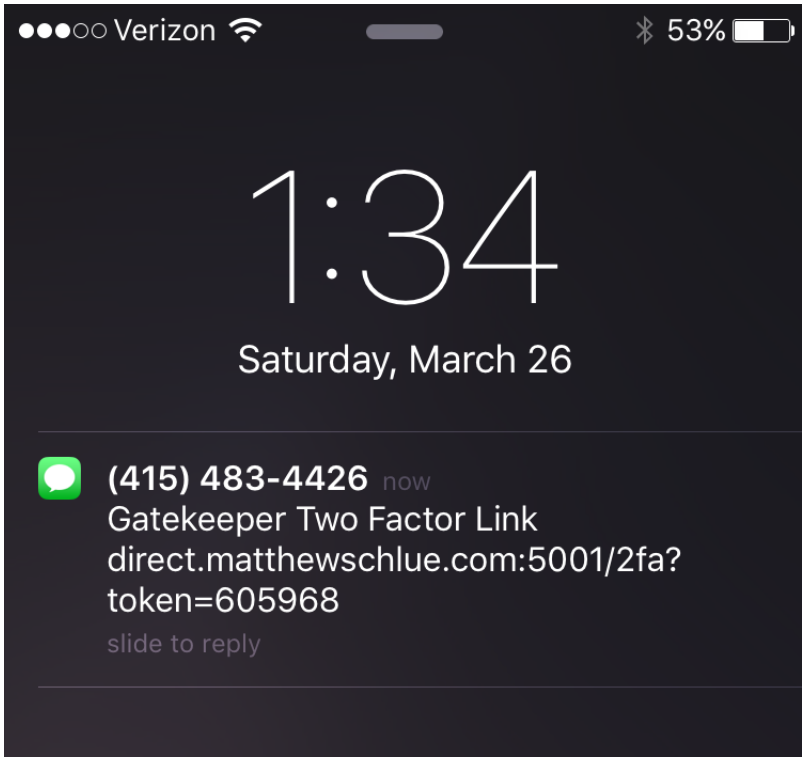

Figure 4.5 Two-Factor Authentication Token

```
WARNING:root:starting web service
Checking client data payload: {u'username': u'matthewschlue', u'ip': u'192.168.1.125',
Client passed the check for username!
Client passed the check for ip!
Client passed the check for mac!
Client failed the check for hostname
Sending two factor challenge
Client failed two factor challenge
127.0.0.1 - - [2016-04-09 18:08:03] "POST /auth HTTP/1.1" 403 248 0.007863
```

Figure 4.6 Client Failing Two-Factor Authentication

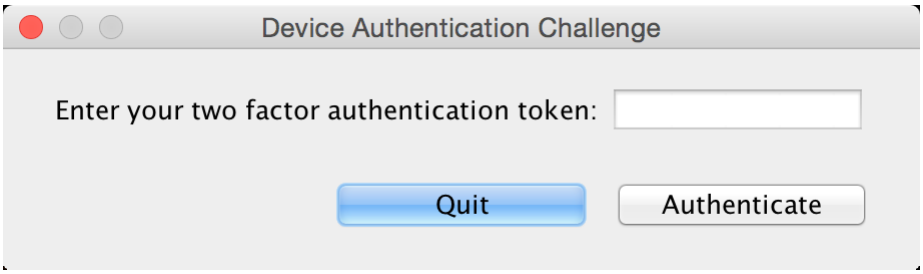

Figure 4.7 HMI Application Two-Factor Prompt

failed during this evaluation a two-factor screen is show to a user before login. After providing the proper code, a user is allowed access to the application.

There are several security advantages that are immediately realized in this experiment when implementing this the active device authentication mechanisms. The authentication server allows the administrator to more robustly log authentication attempts to the HMI applet. A proprietary HMI console won't always contain the relevant logs an administrator will need to access ongoing login attempts to an application.

The data points collected in this application are relatively easy for a skilled hacker to change on their system, however it is very difficult to forge proper device credentials. Initially the attacker may not realize their machine is essentially being man in the middled and tested for device authentication. This may slow down a malicious actor, but seldom will they stop at the first road block. Java decompilers such as the one listed at http://jd.benow.ca/ provide a quick way for the attacker to decompile a Java class file and discover the data being collected. Knowing what system data is being collected isn't the same as knowing the unique values required for device authentication. Now, along with having a valid username and password an attacker also has nine additional pieces of information they need to correctly provide. A brute force attempt against the authentication server, while probable, is very unlikely to succeed before detection. This difficulty is compounded if an attacker has to switch IP addresses after each unsuccessful attempt. In this scenario if the time to detection plus the time to mitigation is less than the time to reverse engineer and brute force the authentication system, we can properly protect the HMI application.

CHAPTER 5. CONCLUSION

5.1 Summary

Current third generation SCADA networks have repurposed outdated and existing systems, placing them on the publicly accessible locations of the Internet or corporate intranets. Chapter 1 elaborates on the significant gaps in the current security posture of these newly connected networks and surveys current security landscape in both SCADA networks and HMI applications. Chapter 2 reviews current security mechanisms, including firewalls and intrusion detection systems, and concludes that these techniques alone are no longer potent enough to stop a malicious actor. Chapter 3 presents *active device authentication*, as a technique that combines an intelligent firewall with HMI application code injection to gather client-side system profile data and facilitate authentication challenge-response logic in legacy HMI applications. Finally, chapter 4 demonstrates the feasibility of *active device authentication* by implementing *Gatekeeper* and modifying the compiled binary of an open source demonstration SCADA application originally distributed as a Java applet.

5.2 Discussion and Future Work

The enhanced approach to security introduced in this paper takes protection a step further by adding data collection mechanisms in the application layer of an Internet facing HMI application. These mechanisms collect and relay unique information about the computer system attempting to access the HMI application. This information is evaluated by a server on the SCADA network against a whitelist of known device profiles. If the connecting system is unable to pass the newly embedded security checks its IP address is not allowed access to the SCADA network. One concern with this approach is the tendency of security to fail closed, while recent

trends in reliable computing (such as SCADA systems) is to fail open. In mission critical situations that added security layer may actually be viewed as a hinderance to the network. We would comment on this view by noting that the action taken by *Gatekeeper* could be modified to simply notify the network operators of the suspicious activity (now detectable as a result of the security layer) and continue to fail open until a decision is made by a human operator.

This paper has proven the ability to increase security on a legacy HMI application by injecting logic for system profiling and facilitating authentication challenge-responses. An important area of work that is left as an open problem in this paper is testing these protection mechanisms in several large scale, live SCADA testbeds. As noted in Chapter 1, SCADA system's depend on high reliability and uptime. While the gatekeeper application proved to add security with demo applications it is important that it is tested in a live environment where any adverse affects can be measured.

BIBLIOGRAPHY

Blakely, B. A. (2012). Cyberprints: Identifying cyber attackers by feature analysis.

Bursztein, E., Benko, B., Margolis, D., Pietraszek, T., Archer, A., Aquino, A., Pitsillidis, A., and Savage, S. (2014). Handcrafted fraud and extortion: Manual account hijacking in the wild. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 347–358. ACM.

Cai, N., Wang, J., and Yu, X. (2008). Scada system security: Complexity, history and new developments. In *Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on*, pages 569–574. IEEE.

Durumeric, Z., Wustrow, E., and Halderman, J. A. (2013). Zmap: Fast internet-wide scanning and its security applications. In *Usenix Security*, volume 2013.

ICS-CERT (2016). Overview of cyber vulnerabilities.

Independence, E. (2007). Security act of 2007. *Public law*, 110(140):19.

Langner, R. (2013). To kill a centrifuge: A technical analysis of what stuxnets creators tried to achieve. *Online: http://www. langner. com/en/wp-content/uploads/2013/11/To-kill-a-centrifuge. pdf*.

Moore, H. D. (2013). Scanning darkly (derbycon 3.0 keynote). In *DerbyCon 3.0*.

MSDN (2016). SSL Protocol Stack.

Pothamsetty, F. (2016). SCADA HoneyNet Project: Building Honeypots for Industrial Networks.

Rehman, R. U. (2003). *Intrusion detection systems with Snort: advanced IDS techniques using Snort, Apache, MySQL, PHP, and ACID*. Prentice Hall Professional.

SANS (2008). Network IDS and IPS Deployment Strategies.

Stouffer, K., Falco, J., and Scarfone, K. (2011). Guide to industrial control systems (ics) security. *NIST special publication*, 800(82):16–16.

Tischer, M., Durumeric, Z., Foster, S., Duan, S., Mori, A., Bursztein, E., and Bailey, M. (2016). Users Really Do Plug in USB Drives They Find. pages 1–14.

Urias, V., Van Leeuwen, B., and Richardson, B. (2012). Supervisory command and data acquisition (scada) system cyber security analysis using a live, virtual, and constructive (lvc) testbed. In *MILITARY COMMUNICATIONS CONFERENCE, 2012-MILCOM 2012*, pages 1–8. IEEE.

Wang, W. and Lu, Z. (2013). Cyber security in the smart grid: Survey and challenges. *Computer Networks*, 57(5):1344–1371.

Yadron, Ziobro, L. (2014). Target Hackers Used Stolen Vendor Credentials.